Semaine d'Etude Maths–Entreprises 1

4–8 avril 2011, Institut Henri Poincaré (Paris)

Deux problèmes sur les parcs solaires

N. Aguillon[a] S. Benzekry[b]
J. Bettinelli[a] P. Bochard[a]
N. Bonnotte[a] G. Delgado[c]
L.-M. Imbert[d] G. Lepoultier[a]
L. Navoret[a] E. Parini[e]

[a] *Laboratoire de Mathématiques d'Orsay, Université Paris-Sud, 91405 Orsay, France*
[b] *LATP, Université de Provence, Marseille, France*
[c] *CMAP, Ecole Polytechnique, 91128 Palaiseau, France*
[d] *Laboratoire Jacques-Louis Lions, UPMC, 75005 Paris, France*
[e] *Ceremade, Université Paris-Dauphine, 75775 Paris, France*

Sujet proposé par :

GDF SUEZ

Correspondant : A. Fuser (GDF–Suez)

Maths & Entreprises

cnrs dépasser les frontières

Résumé

Le premier problème consiste à étudier le placement quasi-optimal de parcs de panneaux solaires dans une région géographique définie. Si on suppose que la puissance d'un parc est directement proportionnelle à sa surface, on voudra maximiser la somme des surfaces des parcs. Néanmoins, la construction de tels parcs solaires est souvent soumise à des contraintes légales portant sur la puissance maximale de chaque parc et sur la distance entre deux parcs. Les idées proposées dans ce rapport tiendront donc compte de ces contraintes.

Le deuxième problème concerne les outils de stockage pour l'optimisation de la vente d'énergie solaire. On suppose que l'on dispose d'un parc de panneaux solaires produisant une certaine puissance électrique en fonction de l'ensoleillement. A chaque instant on peut décider de vendre la totalité de la production au prix courant, ou bien d'en stocker une partie, ce qui occasionne un surcoût, mais peut permettre de différer la vente pour viser un meilleur prix. L'objectif est de calculer le profit maximal que l'on peut réaliser sur une année en connaissant pour chaque heure le prix de vente sur le marché ainsi que l'ensoleillement.

Numéro de publication : SEME001-2011-04-B

Première partie

Placement quasi-optimal de parcs de panneaux solaires

1 Introduction

Soit $\mathrm{K} \subset \mathbb{R}^2$ un ensemble compact. Pour $n \in \mathbb{N}$, $\mathrm{A}_{\max}$, $d > 0$, on s'intéresse à maximiser la quantité

$$\varphi(\mathrm{E}_1, ..., \mathrm{E}_n) = \sum_{i=1}^{n} |\mathrm{E}_i|$$

dans la classe $\mathcal{A}$ des ensembles admissibles :

$$\mathcal{A} = \Big\{ (\mathrm{E}_1, ..., \mathrm{E}_n) \,|\, \mathrm{E}_i \subset \mathrm{K} \text{ fermé}, \, |\mathrm{E}_i| \leqslant \mathrm{A}_{\max}, \ \mathrm{dist}(\mathrm{E}_i, \mathrm{E}_j) \geqslant d \text{ si } i \neq j \Big\}.$$

On utilise la convention $|\emptyset| = 0$, $\mathrm{dist}(\mathrm{E}, \emptyset) = \infty$ pour tout ensemble E.

Le problème est lié au placement optimal de parcs solaires dans une région géographique définie. Si on suppose que la puissance d'un parc est directement proportionnelle à sa surface, on voudra donc maximiser la somme des surfaces des parcs. Néanmoins, la construction de tels parcs solaires est soumise (au moins en France à l'état actuel) à des contraintes légales : la puissance de chaque parc ne peut pas dépasser 12 MW, et la distance entre deux parcs doit être au moins 500 mètres. Il faut observer qu'un « parc », selon notre définition, ne doit pas nécessairement être un ensemble connexe ; donc, la contrainte sur la distance ne doit pas être vérifiée sur les différentes composantes connexes d'un même parc.

2 Existence d'une solution

Dans ce paragraphe on va montrer que le problème admet effectivement une solution.

Proposition. *Il existe un élément* $(\mathrm{P}_1, ..., \mathrm{P}_n)$ *de* $\mathcal{A}$ *tel que*

$$\varphi(\mathrm{P}_1, ..., \mathrm{P}_n) = \sup_{\mathcal{A}} \varphi,$$

pourvu que la classe $\mathcal{A}$ *soit non vide.*

Démonstration. Soit $(\mathrm{E}_1^{(k)}, ..., \mathrm{E}_n^{(k)})$ une suite telle que

$$\varphi(\mathrm{E}_1^{(k)}, ..., \mathrm{E}_n^{(k)}) \to \sup_{\mathcal{A}} \varphi =: \mathrm{M}.$$

Comme l'ensemble K est borné, il existe un élément $(\mathrm{E}_1, ..., \mathrm{E}_n) \in [\mathcal{P}(\mathrm{K})]^n$ tel que, à une sous-suite près, $\mathrm{E}_i^{(k)} \to \mathrm{E}_i$ dans la distance de Hausdorff définie comme

$$d_{\mathrm{H}}(\mathrm{C}, \mathrm{D}) = \inf \{ \varepsilon > 0 \,|\, \mathrm{C} \subset \mathrm{D}^\varepsilon \wedge \mathrm{D} \subset \mathrm{C}^\varepsilon \}$$

où $\mathrm{A}^\varepsilon = \mathrm{A} + \mathrm{B}_\varepsilon$, B_ε étant la boule de rayon ε. Alors :

1. Pour $i \neq j$, $\mathrm{dist}(\mathrm{E}_i, \mathrm{E}_j) \geqslant d$. En effet, pour tout $x_i \in \mathrm{E}_i$ et $x_j \in \mathrm{E}_j$, on trouve des points $x_i^{(k)} \in \mathrm{E}_i^{(k)}$ et $x_j^{(k)} \in \mathrm{E}_j^{(k)}$ (à une sous-suite près) tels que $x_i^{(k)} \to x_i$ et $x_j^{(k)} \to x_j$ (voir [6, Theorem 4.5 a]). Mais alors

$$d \leqslant |x_i^{(k)} - x_j^{(k)}| \leqslant |x_i^{(k)} - x_i| + |x_i - x_j| + |x_j - x_j^{(k)}|,$$

et en faisant tendre k vers l'infini on obtient $|x_i - x_j| \geqslant d$, d'où la thèse ;

2. On a $|\mathrm{E}_i| \geqslant \limsup_{k\to\infty} |\mathrm{E}_i^{(k)}|$.

Pour chaque $i = 1, ..., n$, il est donc possible de prendre des ensembles $\mathrm{P}_i \subset \mathrm{E}_i$ avec $|\mathrm{P}_i| = \limsup_{k\to\infty} |\mathrm{E}_i^{(k)}|$. Comme $|\mathrm{P}_i| \leqslant \mathrm{A}_{\max}$ et $\mathrm{dist}(\mathrm{P}_i, \mathrm{P}_j) \geqslant d$ pour $i \neq j$, on a que $(\mathrm{P}_1, ..., \mathrm{P}_n) \in \mathcal{A}$. On obtient donc

$$\mathrm{M} \geqslant \sum_{i=1}^{n} |\mathrm{P}_i| = \sum_{i=1}^{n} \limsup_{k\to\infty} |\mathrm{E}_i^{(k)}| \geqslant \limsup_{k\to\infty} \sum_{i=1}^{n} |\mathrm{E}_i^{(k)}| = \mathrm{M},$$

d'où

$$\varphi(\mathrm{P}_1, ..., \mathrm{P}_n) = \mathrm{M}.$$

□

3 Approche par algorithmes génétiques (en 1D)

La méthode d'algorithmes génétiques est une technique robuste et puissante qui permet souvent d'obtenir, en un temps « raisonnable » une solution approchée à un problème d'optimisation général que l'on ne sait pas résoudre explicitement par d'autres méthodes standards. Le principe général est le suivant. On cherche à maximiser sur un ensemble E une fonction φ. On se donne (arbitrairement) des paramètres n, $p \in \mathbb{N}$, s, $\mu \in \{0, 1/p, 2/p, \ldots, 1\}$. On choisit selon la méthode de notre choix (souvent aléatoire) p éléments de E qui seront des candidats potentiels à notre problème. Ces éléments forment la première génération d'individus. On procède ensuite de façon itérative, créant à chaque nouvelle génération p nouveaux individus, potentiellement meilleurs que les p précédents, à l'aide de trois principes fondamentaux.

1. Sélection. On choisit les sp meilleurs individus, i.e. ceux pour lesquels φ est maximale ;
2. Mutation. On crée μp individus par mutation : on choisit aléatoirement un individu de la génération précédente (selon la loi de notre choix, typiquement uniforme ou uniforme biaisée par la valeur de φ) et on le modifie légèrement ;
3. Croisement. On crée $(1 - s - \mu)p$ individus par croisement : on choisit deux individus de la génération précédente (là encore, selon la loi que l'on veut) et on crée un nouvel individu.

On s'arrête ensuite à la génération n, et on garde le meilleur individu de cette génération. Tout le problème consiste alors à définir précisément les mécanismes de mutation et croisement, puis à trouver les bon paramètres n, p, s, et μ.

Dans notre problème de placement de parcs, nous nous sommes placés en dimension un pour plus de simplicité dans un premier temps. La méthode que nous avons utilisée devrait fonctionner en deux dimensions, moyennant quelques modifications, mais nous ne l'avons

pas implémenté, faute de temps. Plus précisément, nous nous plaçons sur un sous-ensemble fini K de $\mathbb{N}$, et voyons les configurations de parcs comme des éléments de $\{0,1\}^{\mathrm{K}}$, où 0 signifie qu'il n'y a pas de parc et 1 signifie qu'il y a un parc. On cherche alors à maximiser sur $\{0,1\}^{\mathrm{K}}$ une fonctionnelle du même type que précédemment, faisant apparaître un facteur de pénalisation pour les configurations ne respectant pas les contraintes (en une dimension, on demande aux parcs d'avoir une longueur inférieur à un entier l et d'être espacés les uns par rapport aux autres d'au moins un entier d).

Le mécanisme de mutation que nous avons utilisé consistait à

- avec une certaine probabilité, on choisit (aléatoirement, de façon biaisée par sa taille) un grand parc dans la configuration, et on lui enlève une case à gauche ou droite ;
- avec une certaine probabilité, on choisit (aléatoirement, de façon biaisée par sa taille) un petit parc, et on lui ajoute une case à gauche ou droite (en « traversant » éventuellement les trous de K) ;
- avec la probabilité restante, on choisit (aléatoirement, de façon biaisée par sa taille) un grand espace entre deux parcs, et on ajoute au milieu un nouveau parc d'une case.

Le mécanisme de croisement consistait simplement à choisir deux configurations et en créer une nouvelle en prenant le début de la première, le milieu de la seconde, et la fin de la première, où ces parties étaient déterminées à l'aide de deux entiers tirés uniformément au hasard.

Pour commencer, nous avons simplement mis du parc avec probabilité 1/2 indépendamment sur chaque case. Sur l'exemple illustré ici,

$$\begin{aligned} \mathrm{K} &= \{1,2,3,4,5,8,9,10,14,15,16,18,19,20\}, \\ l &= 3, \\ d &= 4, \end{aligned}$$

et nous avons choisi les paramètres $n = 20$, $p = 100$, $c = 10$, et $\mu = 20$. Les figures 1 à 3 représentent les p configurations empilées les unes sur les autres, classées selon la valeur de φ, les meilleures configurations étant en bas de l'image. Les cases noires représentent les zones interdites (i.e. n'appartenant pas à K) et les cases blanches sont les cases de K inoccupées par des parcs. Enfin, chaque parc est coloré d'une couleur différente pour une meilleure visibilité (le premier en bleu, le second en vert, le troisième en rouge, etc.).

Évidement, à la première génération, les configurations sont très mauvaises (la plupart ne respectent même pas les contraintes), mais on voit sur cet exemple que dès la seconde génération, les configurations sont bien meilleures, et au bout de 20 générations, elles sont très bonnes. La figure 4 représente la taille totale des parcs de la meilleure configuration en fonction de la génération. On voit qu'en à peine 3 générations, on obtient une très bonne configuration.

En deux dimensions, une méthode similaire semble applicable. Un mécanisme de mutation possible serait de faire varier la frontière dans un sens ou un autre ou de créer un petit parc dans une « grande » zone vide, et un mécanisme de croisement envisageable serait de garder une partie de la première configuration et de remplacer le reste par la partie correspondante de la seconde configuration.

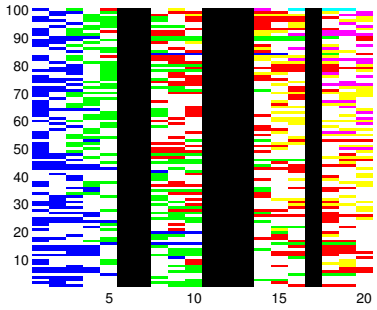

FIGURE 1 – Première génération.

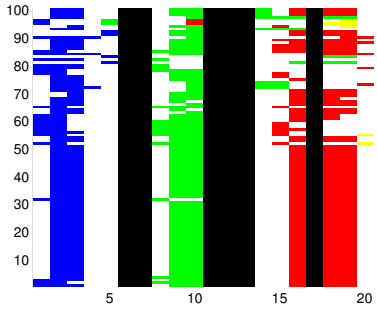

FIGURE 2 – Seconde génération.

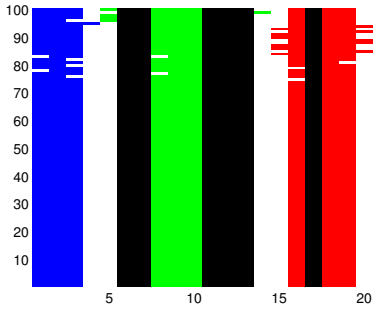

FIGURE 3 – Vingtième génération.

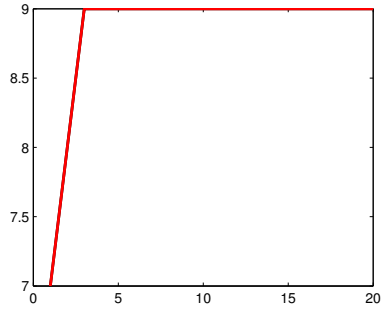

FIGURE 4 – Taille totale des parcs.

4 Formulation variationnelle

4.1 Première formulation

L'idée repose sur la traduction du problème de minimisation initial avec ses contraintes en une seule fonctionnelle faisant intervenir directement les contraintes. On va donc vouloir minimiser la fonctionelle :

$$\mathcal{F} : \begin{cases} \mathcal{P}(\mathrm{D})^n \to & \mathbb{R} \\ (\Omega_1, \ldots, \Omega_n) \mapsto & -\sum_{i=1}^{n} |\Omega_i| + \lambda_1 \sum_{i=1}^{n} (|\Omega_i| - \mathrm{S})^+ + \lambda_2 \sum_{i,j} |\Omega_{i,\frac{d}{2}} \cap \Omega_{j,\frac{d}{2}}| \end{cases}$$

où D est le domaine sur lequel on veut placer des panneaux, les $(\Omega_i)_{i\in\{1,\ldots,n\}}$ sont les parcs de panneaux solaires, S la surface maximale autorisée pour un parc et d la distance minimale devant séparer deux parcs distincts et où

$$\Omega_{i,r} = \{x | d(x, \Omega_i) \leqslant r\}\,.$$

De la sorte, le second terme de la fonctionnelle s'interprète comme une pénalisation de la contrainte $|\Omega_i| \leqslant \mathrm{S}$ alors que le troisième terme pénalise le fait que deux parcs distincts doivent être à une distance au moins d l'un de l'autre. λ_1 et λ_2 sont les poids associés aux différentes contraintes. Remarquons qu'on à ici supposé connu le nombre de parcs à placer, ce qui n'est pas nécessairement le cas en réalité. D'un point de vue algorithmique, une

borne superieure sur le nombre de parcs, qu'on peut facilement obtenir par un argument d'aire suffit : les parcs en trop vont s'atrophier jusqu'à disparaitre.

4.2 Seconde formulation

Une fois le problème ainsi traduit et comme il nous semblait intéressant d'avoir des résultats numériques à exposer à la fin de la semaine, s'est posée la question de la mise en place d'un algorithme de minimisation de la fonctionnelle. Comme celle-ci prend une famille d'ensemble en argument, on n'est pas dans un cas de minimisation classique. Une solution pour remédier à ça a été de reformuler le problème en s'autorisant seulement des parcs circulaires. Des tests sur des cas simples tels un carré nous avaient convaincus que les ensembles optimaux avaient des frontières plutôt droites, on autorise donc les centres de nos cercles à être en dehors du domaine où l'on veut placer des parcs solaires ; en pratique, on devrait ainsi pouvoir construire des parcs dont les frontières sont des portions de cercles ou des quasi-droites lorsque les centres des cercles sont très éloignés des parcs.

On se retrouve maintenant à minimiser la fonctionnelle :

$$\tilde{\mathcal{F}} : \begin{cases} \mathbb{R}^n \times \mathbb{R}^n & \to & \mathbb{R} \\ (p,r) & \mapsto & -\sum_{i=1}^{n} |\Omega_i| + \lambda_1 \sum_{i=1}^{n} (|\Omega_i| - \mathrm{S})^+ + \lambda_2 \sum_{i,j} |\Omega_{i,\frac{d}{2}} \cap \Omega_{j,\frac{d}{2}}| \end{cases}$$

où p_i est le centre du cercle i et r_i son rayon, ce pour quoi on dispose de nombreuses méthodes algorithmiques.

5 Minimisation par recuit simulé

Essentiellement à cause du peu de temps dont nous disposions, nous avons choisi de minimiser notre fonctionnelle par recuit simulé : cette méthode à l'avantage de ne nécessiter que le calcul des $\tilde{\mathcal{F}}(p,r)$ et pas de calcul de gradient qui aurait nécessiter d'expliciter $\tilde{\mathcal{F}}(p,r)$ en fonction des (p,r). Un autre avantage de cette méthode est de ne pas rester bloquer dans des extremums locaux, ce qui est une bonne chose puisque quelques cas simples nous ont convaincus qu'il n'y a pas unicité du minimiseur. Pour mémoire, voici une description de l'algorithme de recuit simulé dans notre cas particulier :

1. On se fixe un domaine rectangulaire R suffisamment grand contenant le domaine ou l'on veut placer des parcs D. Les centres de nos cercles seront dans R. On place un nombre n de cercles, dont les centres sont répartis aléatoirement dans R et dont les rayons sont aléatoires. On se fixe également une température initiale T_0 intervenant dans le precessus de recuit ;
2. On modifie aléatoirement les positions des centres des cercles et leurs rayons (tout en concervant les centres dans le domaine R). On calcule l'énergie associée à la nouvelle configuration. Si celle-ci est plus faible, on conserve la nouvelle position, si elle est plus forte, on la conserve avec une probabilité $e^{-\mathrm{T}_0\Delta}$ où Δ est la différence entre la nouvelle énergie et l'ancienne et dans les autres cas on repart de la position anterieure, puis on itère le processus.

6 De possibles améliorations

Plusieurs améliorations sont possibles :

- tout d'abord, on peut s'autoriser d'autres formes que des cercles comme « briques élémentaires » : on peut penser à des ellipses, des carrés, etc. ;
- en ce qui concerne la méthode de minimisation, les domaines donnés sur lesquels on veut placer des parcs solaires sont en pratique des polygones ; il est donc possible, bien que fastidieux, d'expliciter $\tilde{\mathcal{F}}(p, r)$ en fonction de (p, r) puis de calculer son gradient pour appliquer une méthode de type descente de gradient bien qu'il ne soit pas clair qu'une méthode naïve de ce type soit plus efficace à cause du fait de l'existence de minimums locaux dans lesquels on resterait bloqué. En fait il serait probablement plus efficace de conjuguer méthode de gradient et recuit simulé en utilisant le gradient pour avoir une bonne direction de descente tout en s'autorisant aléatoirement à sortir de minimums pour voir si ceux-ci sont locaux ;
- dans nos exemples le choix des paramètres λ_1,λ_2 et T_0 s'est fait par tatonement ; des utilisateurs attendraient sûrement que le choix soit automatique, ce qui peut également être inclu dans l'algorithme de minimisation ;
- enfin même si la méthode est naïve, elle peut être utile pour initialiser la méthode plus subtile par déformation de formes exposée ensuite.

7 Méthode *level-set*

Dans chaque composante connexe K_i de K, on va dessiner les différentes parties de chaque parc P_j. Chaque parc sera paramétré par une fonction *level-set* φ_j en suivant l'idée de Osher et Sethian (voir [5]). On va résoudre l'équation de transport de chaque φ_j avec une vitesse d'advection V telle que optimise un critère pertinente.

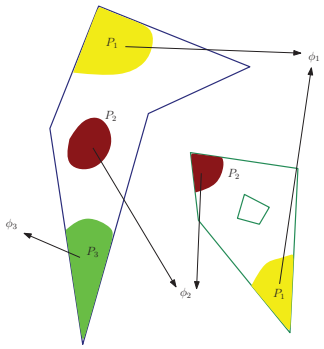

FIGURE 5 – Stratégie d'optimisation avec *level-sets*.

Cette approche permets représenter de manière implicite la géométrie de chaque parc et traiter le problème d'évolution de la surface P_j à travers une EDP, en permettant la génération des changements topologiques (fusion de deux parties connexes d'un parc). On définit la fonction ligne de niveau φ_j sur K comme

$$\begin{cases} \varphi_j(x) = 0 \Leftrightarrow & x \in \partial \mathrm{P}_j \cap \mathrm{K}, \\ \varphi_j(x) > 0 \Leftrightarrow & x \in \mathrm{P}_j, \\ \varphi_j(x) < 0 \Leftrightarrow & x \in (\mathrm{K} \setminus \bar{\mathrm{P}}_j). \end{cases}$$

La normale extérieure n_j à P_j est récupérée comme

$$n_j = \frac{\nabla \varphi_j}{|\nabla \varphi_j|}$$

et la courbure H_j est donnée par la divergence de la normale, $\mathrm{H}_j = \operatorname{div} n_j$. Il faut remarquer que même si n_j et H_j sont définies au niveau théorique seulement sur $\partial \mathrm{P}_j$, la ligne de niveau permet de définir facilement son extension sur tout le domaine K.

En suivant le processus d'optimisation, la forme de chaque P_j va évoluer selon un temps fictif ce qui correspond au pas de descente. C'est bien connu que si la forme évolue en temps, l'évolution de la ligne de niveau est gouvernée par une équation du type Hamilton-Jacobi (voir [4]). Pour être plus précis, on va assumer que la forme $\mathrm{P}_j(t)$ évolue en temps $t \in \mathbb{R}$ avec vitesse normale $\mathrm{V}(x,t)$. Alors

$$\varphi_j(t, x(t)) = 0 \quad \text{pour tout } x(t) \in \partial \mathrm{P}_j(t).$$

En calculant la dérivée totale par rapport au temps on arrive à

$$\frac{\partial \varphi_j}{\partial t} + \dot{x}(t) \cdot \nabla \varphi_j = \frac{\partial \varphi_j}{\partial t} + \mathrm{V} n_j \cdot \nabla \varphi_j = 0.$$

Comme $n_j = \frac{\nabla \varphi_j}{|\nabla \varphi_j|}$ on va obtenir

$$\frac{\partial \varphi_j}{\partial t} + \mathrm{V} |\nabla \varphi_j| = 0.$$

Cette équation d'Hamilton-Jacobi est défini sur tout le domaine K, et non seulement sur la frontière $\partial \mathrm{P}_j$, si l'on connait la vitesse V partout. C'est bien connu que les équations d'Hamilton Jacobi n'admettent pas normalement des solutions régulières. L'existence et unicité peuvent être obtenus dans le cadre des solutions de viscosité (voir [2]).

Au moment de choisir la vitesse d'advection V, on utilise une méthode du type descente à pas constant avec $\mathrm{V} = \theta \cdot n_j$ tel que $\mathrm{J}'(\{\mathrm{P}_j\}_j)(\theta) < 0$, ou J' est la dérivée de forme de J par rapport à chaque P_j (voir définition en [3]) et J est le critère pénalisé à maximiser définie comme

$$\mathrm{J}\left(\{\mathrm{P}_j\}_{j=1..n}\right) = \sum_j \int_{\mathrm{P}_j} d\mathrm{V} - \lambda_1 \sum_j \left(\left(\int_{\mathrm{P}_j} d\mathrm{V} - \mathrm{A}_{max} \right)^+ \right)^2 - \lambda_2 \sum_{j,k;j\neq k} \int_{\partial \mathrm{P}_j} ((d_{\mathrm{P}_k}(x - d_{\mathrm{off}} n_k(x)))^+)^2 ds(x).$$

Le premier terme corresponde à l'aire totale des parcs solaires, et le deuxième terme (avec un multiplicateur de Lagrange $\lambda_1 > 0$) représente la contrainte d'aire maximale (A_{max}) de chaque parc et finalement le troisième terme (avec multiplicateur de Lagrange $\lambda_2 > 0$) la contrainte de distance minimale (d_{off}) entre deux parcs P_j et P_k décrite en [1].

A continuation un exemple numérique très simple sur un domaine K carré, avec deux parcs solaires, un d'eux (inférieur) non-connexe. On peut apprécier dans la figure finale (itération numéro 20) que la méthode de *level-set* est capable de fusionner deux composantes connexes du même parc et garder la distance minimale d_{off} entre parcs différents.

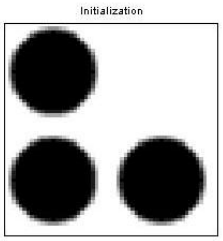

FIGURE 6 – Initialisation

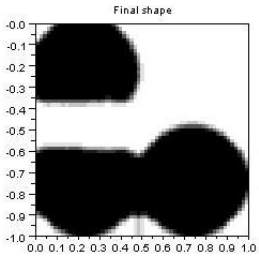

FIGURE 7 – Structure après 20 iterations.

Au même temps, on peut déduire à partir de ces résultats que l'algorithme a la tendance d'étendre chaque parc de manière uniforme (en respectant les contraintes) sans produire mouvements plus compliqués (de rotation par exemple) par rapport à la configuration initiale. Ce phénomène entraine une forte dépendance de la configuration optimal des parcs à partir de la configuration initiale, ce qui rendre l'algorithme pas très robuste.

Pour contourner cette difficulté, une première amélioration à la méthode corresponde à suivre l'approche multi-phase (chaque phase un parc) de [7], ou la fonctionnelle d'énergie à minimiser est décrite par la tension superficielle (proportionnelle à la longueur de l'interface) plus une énergie volumique (proportionnelle à l'aire de chaque phase). Dès qu'on arrive à la configuration optimale, on applique la contrainte de distance minimale dont nous avons parlé toute à l'heure. Une deuxième idée c'est d'utiliser un algorithme de direction faisable ([8]) au lieu d'un algorithme de gradient avec pénalisation, en laisant plus de liberté de mouvement à chaque parc.

8 Remise en perspective

Le résumé a jusqu'à maintenant choisi de s'articuler autour des solutions les plus probantes, mettant de coté les idées les ayant fait germer. Nous allons maintenant aborder les choses sous un angle un peu plus chronologique et nous intéresser à deux choses :

- les premières idées, qui, bien que bonnes à jeter, ont permis une meilleure compréhension du problème et d'avancer ;
- les pistes misent de coté : soit elles nous semblaient douteuses, soit le temps nécessaire pour les creuser et juger de leur viabilité était insuffisant.

Ceci afin de remettre en perspective les deux idées centrales (recuit simulé et *level-set*).

L'approche originale, qui ne se voulait absolument pas une solution, mais une première tentative pour se faire la main sur le problème, consistait à regarder un domaine connexe, placer un certains nombres de points et « gonfler » à partir de ces points (par exemple par une marche aléatoire avec pondération de la proba par la place que fait perdre la case). Un cas simple nous permet de comprendre pourquoi cela ne fonctionne absolument pas et de prendre conscience de certaines caractéristiques du problème. Si l'on traite le carré, on aura tendance à choisir les points de départ extrêmaux (à distance maximale), donc par exemple deux points dans des coins opposés. Ce qui dans certaines configurations, ne va pas permettre d'attraper un bon résultat, puisque la situation finale que l'on cherche, c'est deux parcs parallèles au bord. Cette démarche mit en exergue deux points clés :

– le comportement chaotique du problème, puisqu'une même forme de domaine peut générer des situations différentes et complexes ;
– le problème de la gestion du nombre de parcs qui doit être souple, car on ne sait pas exactement combien nous allons en placer.

Aussi, il fut décidé de chercher des méthodes permettant de gérer une certaine approximation sur le nombre de parcs (qu'on estime grossièrement), permettant par exemple d'en supprimer si on en met trop. Et des méthodes ne figeant surtout pas les choses : il fallait vraiment garder une grande souplesse pour permettre aux parcs de bouger et de changer de forme afin de prendre en compte cet aspect chaotique du problème. Ce sont ces différents paramètres qui ont amené l'approche recuit-simulé ou *level-set*, voir de combiner les deux : le recuit simulé qui permet de gérer une approximation sur le nombre de parcs (on en place plus, on laisse l'algo les sortir du domaine effectif), et une fois que l'algorithme a fourni une première approximation, on peut affiner les choses avec la méthode de *level-set*.

Pour garder cette souplesse, une approche était de regarder les choses sous l'angle d'un problème d'évolution : tenter de modéliser les parcs comme se repoussant suivant une force de type électrique et s'attirant suivant une force de gravitation, les petits parcs voulant fusionner entre eux et les grands se repousser. La chose connue la plus proche est la chimiotaxie, mais c'est encore assez différent. Par contre plutôt que de creuser cette piste, on peut garder l'idée mais l'aborder sous un angle théorique bien différent : un jeu de la vie.

9 Jeu de la vie

Cette méthode permettrait théoriquement de gérer le nombre de parcs (qui peut évoluer/régresser) et leur positionnement, par contre sa « stabilité » et sa faisabilité machine est à voir. Elle se définit directement sur un domaine discrétisé (encore qu'il doit bien être possible de la formuler théoriquement sur un domaine continu). Elle consiste à regarder chaque parcs comme une espèce de bactérie, chacune de ces espèces voulant se développer (avec une taille maximale donnée par les contraintes sur le parc) et émettant une « radioactivité » (variant suivant la taille et la localisation de l'espèce) dans un rayon prescrit par la distance minimale entre les parcs, rendant délicate la reproduction des autres espèces qui y sont soumises, voir les tuant.

Pour cela, on commence par mailler le domaine (pour l'instant, on ne réfléchit pas à de l'adaptation de maillage, on reste sur un maillage fixe. Mais l'idée finale est de faire de l'adaptation de maillage, en suivant finement le front des parcs et libérant de la place ailleurs). Chaque maille aura les quantités suivantes, en plus des informations standards liées au maillage, qui y sont stockées :

– un entier indiquant le parc éventuel d'appartenance : 0 indique qu'aucun parc n'est présent sur la maille, 11 que la maille appartient à la première composante connexe du parc 1, 12 à la deuxième composante connexe du parc 1, etc.,
– une probabilité de reproduction : une case à la frontière de la population a une certaine probabilité de contaminer ses voisines ;
– une probabilité de mort : une case contenant de la population a une certaine probabilité de voir ses membres mourir ;
– une probabilité de naissance : une case inoccupée a une faible probabilité de voir une naissance spontanée se produire. Cette probabilité a pour vocation de peupler

les éventuels zones désertiques que l'algorithme n'aurait pas encore colonisé ou qu'il risquerait de ne pas atteindre ;
- un plus de cela, on stocke par composante connexe l'aire qu'elle occupe, ainsi que l'aire qu'elle exclue.

Ces probabilités reposent sur une quantité qu'il ne semble pas nécessaire de stocker en machine, la radiation. Elle se définit en deux étapes. Premièrement, on définit la radiation associée à la composante connexe d'un parc. Il s'agit d'une quantité qui va influer toutes les autres espèces de bactérie se trouvant à moins de 500 mètre (la distance minimale entre les parcs), de manière progressive. Cette quantité est aussi pondérée par la « marge de manoeuvre » d'une colonie : ainsi une composante assez grosse mais tassée dans un coin, aura tendance à « taper » très fort dans sa seule direction de croissance possible, tandis qu'une colonie au centre pourra se permetre de mourir un petit peu d'un coté pour aller coloniser de l'autre. Formellement, cela peut donner quelque chose comme :

$$\mathrm{rad}_i(x) = \begin{cases} 0 \quad \text{si } d\left(x, \mathrm{Parc}_i\right) \geqslant 500, \\ \dfrac{1}{1 + d\left(x, \mathrm{Composante}_i\right)} \left(\dfrac{\mathrm{Aire}\left(\mathrm{Composante}_i\right)}{\mathrm{Aire(zone\ d'exclusion)}} \right) \quad \text{sinon.} \end{cases}$$

Ensuite, on définit $\mathrm{rad}(x)$ comme la somme de ces quantités sur toutes les composantes de parcs, sauf les composantes du parc auquel appartient x. Notons que cette formule a un sens si on remplace x par une cellule K. Précisons aussi que cette formule n'est qu'une idée et doit être calibrée par la pratique : il faut très probablement ajuster des constantes, voir changer la forme en rajoutant par exempe une puissance pour varier la vitesse de variation ou une exponentielle, etc.

On peut maintenant définir les probabilités. Pour cela, on se donne trois probabilités de référence $\mathrm{P}_n, \mathrm{P}_r, \mathrm{P}_m$ qui correspondent respectivement à la naissance, à la reproduction et à la mort. Typiquement, on choisit P_n assez petit (la naissance spontanée est assez rare), P_r égale à 1 (si rien ne la bloque, la reproduction est automatique). Ensuite, les probabilités sont calculés en fonction de la situation, par exemple :

- la probabilité de naissance d'une cellule vierge K est donnée par la formule $\mathrm{P}_n/(1 + \mathrm{rad(K)})$. Ainsi, plus la radioactivité est élevée, moins la naissance est probable. Profitons en pour dire que si une cellule nait, il convient de la rattacher à un parc. Le parc le plus proche ou le plus petit semble être un choix pertinent (sous réserve que la taille maximale n'ait pas été atteinte !), mais il n'est pas évident que ça ne soit pas toujours la meilleure option et que cela n'altère pas profondément l'algorithme ;
- la probabilité de reproduction d'une cellule à la frontière d'un parc est donnée par $\mathrm{P}_r/(1 + \mathrm{rad(K)})$ (une radioactivité élevée freine la reproduction) si le parc n'a pas atteint sa masse maximale. Si le parc a atteint sa masse maximale, cette probabilité est nulle ;
- la probabilité de mort d'une cellule occupée est $\mathrm{P}_m \mathrm{rad(K)}/(1 + \mathrm{rad(K)})$. Ainsi, s'il n'y a aucune radioactivité, aucune raison de mourir, mais si celle-ci est élevée, on est très proche de P_m. On peut donc supposer qu'une variable bien ajustée de P_m serait proche de 1.

De même que pour la radiation, ces formules mettent en avant un caractère (croissance/décroissance, etc) des probabilités, mais il n'est absolument pas dit que ces formules soient les meilleures possibles.

Ensuite l'idée générale de l'algorithme est assez simple. On se donne un critère d'arrêt

(une certaine stabilité des populations par exemple), tant que ce critère d'arrêt n'est pas réalisé, on boucle sur un balayage du maillage, on teste les naissances, reproductions et morts, on met à jour les probas, et on recommence. L'initialisation se fait soit sur un domaine vide et on laisse faire la vie, soit alors on place quelques parcs suivant notre intuition. Si l'on admet qu'on a des ressources de calculs infinis et qu'on s'autorise un algorithme absolument pas optimisé (donc balayage du maillage autant de fois qu'on le souhaite, etc), il reste quelques questions pratiques, notamment :

- la création d'une nouvelle composante connexe : si un parc est au bord d'une des composantes connexes du domaine, il faut gérer le cas où il « saute » de l'autre coté. On peut pour cela rejoindre artificiellement les mailles d'un coté à celle de l'autre dans la machine. Mais cela soulève quelques petits détails techniques ;
- la fusion de composante connexe : il peut tout à fait arriver en théorie que deux composantes d'un même parc en arrive à fusionner. Il faut donc gérer ce cas en machine ;
- établir le test d'arrêt ;
- la stabilité de l'algorithme : intrinsèquement, il est instable. Par exemple, il doit permettre que deux parcs diagonalement opposé dans un carré s'aligne à l'horizontal, donc traverse une période turbulente. Il faut donc garantir en calibrant bien toutes les données/fonctions qu'une fois traversées ces turbulences, on ne ressorte pas facilement de la situation pour retrouver une situation très mouvante.

Si en plus de gérer ces problème, on veut en plus commencer à être performant en machine (ou même raisonnable), les choses se gatent sérieusement. En effet les critères suivant sont primordiaux et pas évident à traiter pour des personnes n'ayant pas la culture du « code » :

- limiter au maximum le nombre de balayage du maillage, très onéreux en temps ;
- le calcul de la distance entre les ensembles doit probablement couter cher (quoi qu'il semble que le fast-marching permette de traiter ce genre de problème de manière moins onéreuse que prévue) ;
- les calculs d'aire à chaque itération nécessite de bien conserver les aires à chaque étape pour ne pas tout recalculer. Par ailleurs, pour éviter le recours à l'aire d'exclusion, un peu délicate à calculer, on peut peut-être privilégier un critère faisant appel à la frontière du domaine ;
- limiter au maximum les tests.

Deuxième partie

Outils de stockage pour l'optimisation de la vente d'énergie solaire

1 Présentation du problème

Notre problème est le suivant. On dispose d'un parc de panneaux solaires, produisant une certaine puissance électrique en fonction de l'ensoleillement. Pour chaque heure, il est possible de vendre l'énergie ainsi produite au prix du marché, de la stocker sous la forme d'air comprimé dans une cavité souterraine de grande contenance, ou bien au contraire d'utiliser cet air comprimé pour produire plus d'électricité que l'on vend aussi. Quel profit maximal peut-on faire sur une année, si l'on connaît pour chaque heure le prix de vente sur le marché et l'ensoleillement, donc la production d'énergie solaire ?

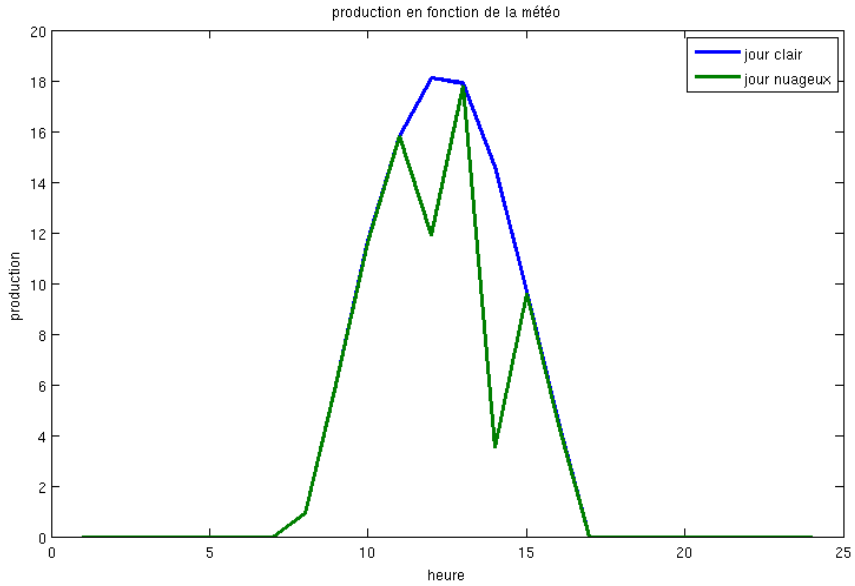

FIGURE 8 – Puissance produite sur une journée

Notons p_x^n le prix à l'heure t^n. La production solaire sera notée prod^n ; elle est connue à l'avance. On note E^n le profit total fait jusqu'au début de l'heure t^n, et V^n le volume d'air comprimé dans la cuve au début de la même heure. Une première contrainte est que l'on doit avoir $\text{V}^n \leqslant \text{V}_{\max}$, avec $\text{V}_{\max} = 300\,000\ \text{m}^3$.

Si l'on décide d'utiliser une puissance p pour comprimer de l'air, et que la cuve contient déjà un volume V, alors ce même volume va augmenter de $c(\text{V}) \cdot p$. Cela va aussi coûter C_c euros par mégawatt ainsi stockés. Attention toutefois : il faut que cette puissance p dédiée à la compression soit au moins supérieur ou égal à 5 MW pendant au moins 2 heures consécutives, pour ne pas endommager le matériel. Et naturellement, il faut que p soit supérieure à la puissance électrique produite par les panneaux solaires... Si l'on note α^n la proportion de puissance solaire consacrée à la compression, si $\alpha^n > 0$, on doit donc

avoir

– soit $\alpha^{n-1}\text{prod}^{n-1} \geqslant 5$ MW et $\alpha^{n}\text{prod}^{n} \geqslant 5$ MW ;
– soit $\alpha^{n}\text{prod}^{n} \geqslant 5$ MW et $\alpha^{n+1}\text{prod}^{n+1} \geqslant 5$ MW.

Si l'on décide au contraire de décomprimer de l'air alors que la cuve contient un volume V d'air comprimé, alors il est possible de produire au maximum une puissance pdc(V), le volume dans la cuve diminue alors de $d(\text{V})$, mais cela coûte C_d euros par mégawatt ainsi produit. Notons β^n la part de cette puissance que l'on décide de récupérer effectivement.

Puisque l'on peut soit comprimer, soit décomprimer, il faut $\alpha^n \cdot \beta^n = 0$. L'évolution du profit et du volume dans la cuve est alors donnée par :

$$\text{E}^{n+1} = \text{E}^n + \underbrace{p_x^n \,\text{prod}^n}_{\text{vente directe}} + \underbrace{\beta^n \,\text{pdc}(\text{V}^n)\,(p_x^n - \text{C}_\text{d})}_{\text{vente issue du stock}} - \underbrace{\alpha^n \,\text{prod}^n\,(p_x^n + \text{C}_\text{c})}_{\text{coût du stockage}}$$

$$\text{V}^{n+1} = \text{V}^n + \underbrace{c(\text{V}^n)\alpha^n \,\text{prod}^n}_{\text{compression}} - \underbrace{\beta^n \, d(\text{V}^n)}_{\text{décompression}}$$

Quelques chiffres :

– le prix moyen est de 80 ?/MWh ;
– le prix maximum est de 1 600 ? ;
– il faut en moyenne 1 200 h pour rempire entièrement la cuve ;
– ... mais il suffit de 50 h pour la vider !

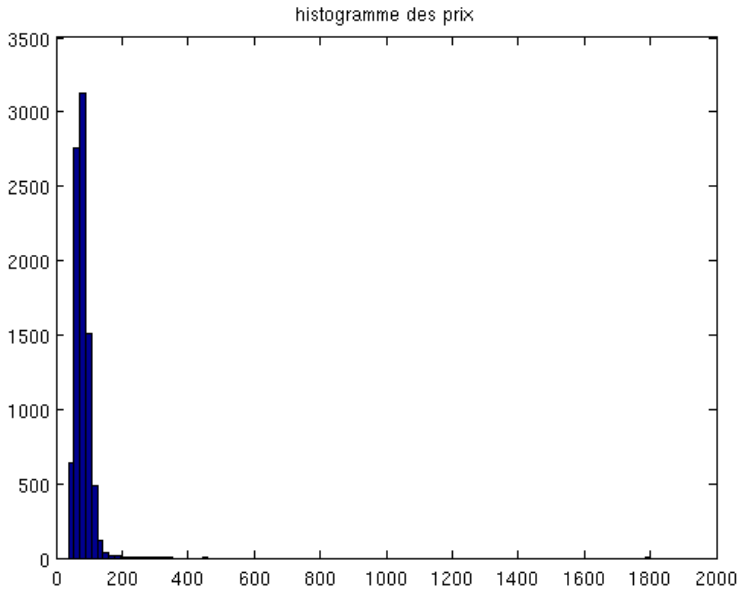

FIGURE 9 – Histogramme des prix

2 Optimisation sans stratégie

Pour calculer le profit maximal possible en utilisant le stockage de l'énergie, la première méthode proposée utilise les *algorithmes génétiques*. Les méthodes génétiques semblent par-

ticulièrement adaptés à notre problème car c'est une méthode d'optimisation permettant de considérer des fonctionnelles ayant de nombreux maxima locaux.

Pour simplifier la méthode, nous considérons que les taux de compressions α^n et décompressions β^n prennent des valeurs discrètes, par exemple $\alpha^n, \beta^n \in \{0, 1\}$. Nous introduisons la variable de choix

$$\gamma^n = \alpha^n - \beta^n = \begin{cases} \alpha^n & \text{si } \alpha^n \geqslant 0, \\ -\beta^n & \text{si } \beta^n \geqslant 0, \end{cases}$$

à valeurs dans $\{-1, 0, 1\}$: elle nous permet de prendre en compte directement la contrainte $\alpha^n \, \beta^n = 0$ et de ne travailler que sur un seul paramètre.

Description de l'algorithme génétique. L'algorithme commence par le tirage aléatoire d'une famille de paramètres de taille $\mathrm{N_{pop}}$, chacun de ces paramètres étant constitué de deux vecteurs de tailles $\mathrm{N} = 8760$ donnant les taux de compression et les taux de décompression à chaque heure durant une année :

$$\mathrm{P}^0 = \left\{ \gamma_i \in \{0, 1\}^{\mathrm{N}} \text{ pour } i \in \{1, \ldots, \mathrm{N_{pop}}\} \right\},$$

puis consiste à faire évoluer cette famille de paramètre itérativement de génération en génération. Étant donnés trois entiers $(\mathrm{N_{meilleurs}}, \mathrm{N_{enfants}}, \mathrm{N_{mut}}) \in \mathbb{N}^3$ tels que $\mathrm{N_{pop}} = \mathrm{N_{meilleurs}} + \mathrm{N_{enfants}} + \mathrm{N_{mut}}$ et une génération P^m au temps m, la génération P^{m+1} au temps $m + 1$ est obtenue par les trois procédés suivants :

1. *Sélection* des meilleurs jeux de paramètres ; après calcul des profits associés à chaque jeux de paramètres, les $\mathrm{N_{meilleurs}}$ meilleurs jeux de paramètres sont conservés ;
2. *Croisement* des jeux de paramètres pour en obtenir de meilleurs ; $\mathrm{N_{enfants}}$ jeux de paramètres sont obtenus de la manière suivante : nous tirons aléatoirement deux jeux de paramètres, selon une certaine loi de probabilité, et après avoir tirer aléatoirement un lieu de coupure, nous raccordons le début de l'un des jeux de paramètres avec la fin du second ;
3. *Mutations* de certains jeux de paramètres : nous tirons aléatoirement $\mathrm{N_{mut}}$ jeux de paramètres, selon une certaine loi de probabilité, et nous modifions aléatoirement la moitié de leurs valeurs.

La loi de probabilité utilisée pour la sélection des jeux de paramètres est une loi linéairement dépendante de leur rang (une fois les jeux de paramètres triés suivant les profits obtenus).

Au cours de l'algorithme, le profit maximal obtenu sur la population des jeux de paramètres,

$$\max_{i \in \{1, \ldots, \mathrm{N_{pop}}\}} \mathrm{E}(\gamma_i),$$

est croissant au cours des générations.

Traitement des contraintes. Le problème de minimisation est assortie de plusieurs contraintes sur les paramètres.

1. Concernant la première contrainte $\alpha^n \beta^n = 0$, elle est implicitement satisfaite en travaillant sur la variable $\gamma^n = \alpha^n - \beta^n$;

2. Concernant les deux contraintes sur la compression, $\alpha^n \text{prod}^n > 5$ et

$$(\alpha^{n-1} = 0 \text{ et } \alpha^n > 0) \Rightarrow \alpha^{n+1} > 0,$$

nous avons choisi de chercher directement les jeux de paramètres satisfaisant ces contraintes. Nous avons donc « projeter » les jeux de paramètres tirer aléatoirement sur l'ensemble des paramètres admissibles. Les jeux de paramètres sont parcourus itérativement et dès qu'une compression est interdite, alors :
   - si c'est la contrainte $(\alpha^n \text{prod}^n > 5)$ qui est mise en défaut, on lui substitue aléatoirement une décompression $\gamma^n = -1$ ou aucune action $\gamma^n = 0$,
   - si c'est la contrainte $(\alpha^{n-1} = 0) + (\alpha^n > 0) \Rightarrow (\alpha^{n+1} > 0)$ qui est mise en défaut, alors on modifie aléatoirement γ^n ou γ^{n+1} (si c'est possible) ;

3. Concernant les contraintes sur le volume, seuls les valeurs du profit E sont affectées tandis que les paramètres de compression/décompression sont inchangés. Il se pourrait toutefois que α^n ne corresponde plus au pourcentage réel de compression et que le pourcentage réel de compression ne vérifie pas $(\alpha^n \text{prod}^n > 5)$. Nous avons toutefois, dans un premier temps, ignorer ce cas.

Pour traiter toutes ces contraintes, nous aurions pu choisir de pénaliser les valeurs de paramètres non-admissibles. Cependant, l'espace des paramètres admissible étant restreint, la probabilité de tirer un jeux de paramètres non-pénalisé est faible ce qui ralentirait la convergence de l'algorithme.

Nous avons fait tourner l'algorithme avec les paramètres suivant : $\text{N}_{\text{pop}} = 60$, $\text{N}_{\text{meilleurs}} = 10$, $\text{N}_{\text{enfants}} = 40$, $\text{N}_{\text{mut}} = 10$. La figure 10 (à gauche) montre l'évolution du profit maximal au cours des générations.

Nous avons considérer ensuite une variante de l'algorithme qui consiste à effectuer la projection sur l'espace des paramètres admissibles seulement toutes les 10 générations. On constate sur la figure 10 (à droite), que l'évolution du profit est plus rapide que précédemment et que le profit maximal atteint au bout de 200 générations est plus élevé. Pour le jeux de paramètres optimal de la dernière génération, le profit maximal obtenu est $5,97$ M ?.

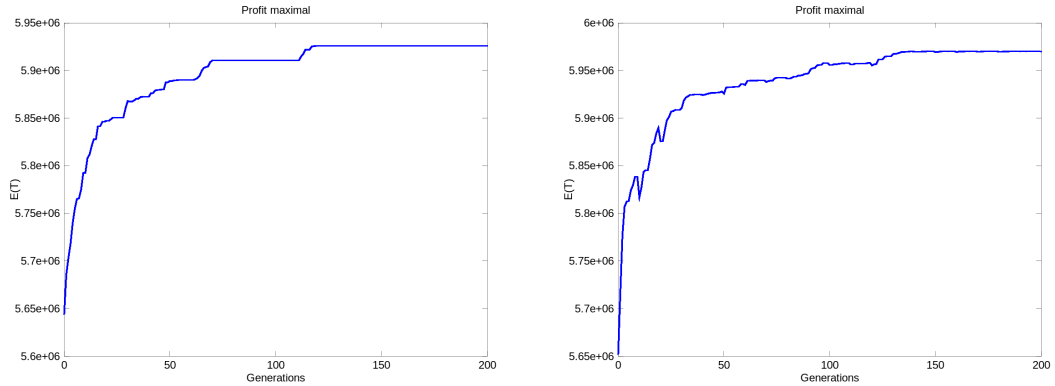

FIGURE 10 – Evolution du profit maximal au cours des générations : à gauche, avec projection à chaque génération, à droite avec projection toutes les 10 générations.

Critère d'arrêt et vitesse de convergence. Ces algorithmes génétiques soulèvent plusieurs questions liées à leurs vitesses de convergence, notamment en fonction des différents paramètres ($\mathrm{N_{pop}}, \mathrm{N_{best}}, \mathrm{N_{enfants}}, \mathrm{N_{mut}}, \ldots$) : le choix des paramètres, pour le moment effectué arbitrairement, pourra être optimisé. Ceci a un impact également sur la nature du critère d'arrêt de l'algorithme : une fois que l'algorithme devient stationnaire, à partir de quand peut-on considérer que le maximum (local) obtenu est proche du maximum global.

Remarquons qu'à temps de calcul fixé, une méthode pour améliorer le résultat serait de débuter avec des jeux de paramètres « à grand profit », sélectionnés par l'une des méthodes avec stratégie décrite dans les paragraphes suivants. Nous pouvons même *postuler* que tout jeux de paramètres obtenu via une stratégie pourra être amélioré grâce à l'algorithme génétique. Les jeux de paramètres ainsi améliorés ne pourront certainement plus être décrit comme résultat d'une stratégie.

La méthode ne permet pas d'en déduire directement des stratégies « en temps réel » de compression-décompression. Cependant, une analyse des probabilités empiriques des compressions et décompressions pourrait peut-être permettre d'établir des stratégies.

3 Optimisation avec stratégie

Une façon naturelle de procéder pour maximiser le profit sur une année est de déterminer, au moment t^n, la stratégie à adopter en fonction de quelques paramètres (ensoleillement, prix de l'énergie dans les prochaines heures, taux de remplissage de la cuve, etc.).

3.1 Méthode à un seuil

La plus simple de ces stratégies consiste à fixer un seuil S, puis de décider à chaque temps t^n de la marche à suivre en fonction de la valeur du prix p_x^n, comparée à ce seuil :

1. Si $p_x^n < \mathrm{S}$, le prix est trop faible et l'on décide donc de stocker l'énergie, à condition toutefois que la production soit suffisante, à la fois au temps t^n mais aussi au temps t^{n+1}, à moins que l'on ait déjà decidé de stocker au temps t^{n-1}. Et bien sûr à condition qu'il reste de la place dans la cuve. Dans ce cas, $\alpha^n = 1$ et $\beta^n = 0$;
2. Si $p_x^n < \mathrm{S}$, mais que l'on ne peut pas stocker, alors on se contente de vendre l'énergie produite. C'est-à-dire que l'on prend $\alpha^n = 0$ et $\beta^n = 0$;
3. Si $p_x^n \geqslant \mathrm{S}$, cela signifie que le prix est intéressant, donc qu'il faut vendre. On destocke donc de l'énergie, ce qui donne $\alpha^n = 0$ et $\beta^n = 1$.

Une optimisation rapide semble indiquer que la recette est maximale pour un seuil $\mathrm{S} = 120$?, et atteint alors $7,4$ M ?.

3.2 Méthode à deux seuils

Un raffinement naturel de la précédente méthode consiste à avoir deux seuils : un seuil de compression S_c, qui représente le montant en dessous duquel il vaut mieux stocker l'énergie, et un seuil de décompression S_d, qui à l'inverse représente le montant au dessus duquel il est intéressant de vendre l'énergie stockée :

1. Si $p_x^n < \mathrm{S}_c$, on stocke l'énergie, si cela est possible, et alors $\alpha^n = 1$ et $\beta^n = 0$;

2. Si $p_x^n < \mathrm{S}_c$, mais que l'on ne peut pas stocker, alors on en fait rien, et l'on prend $\alpha^n = 0$ et $\beta^n = 0$;
3. Si $p_x^n \geqslant \mathrm{S}_d$, on vend de l'énergie stockée, et donc $\alpha^n = 0$ et $\beta^n = 1$.

Dans ce cas, une optimisation rapide donne une recette maximale de $7,4$ M ?, pour des seuils $\mathrm{S}_c = 120$? et $\mathrm{S}_d = 121$? (cf. figures 11 et 12).

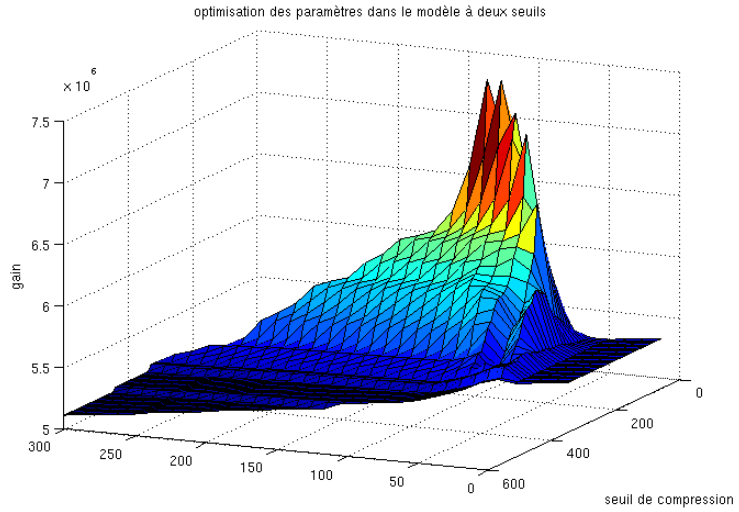

FIGURE 11 – Recette sur l'année en fonction pour deux seuils (3D)

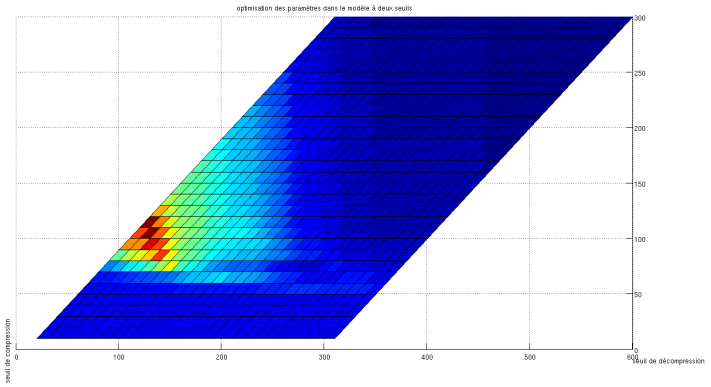

FIGURE 12 – Recette sur l'année en fonction pour deux seuils (projection)

3.3 Première amélioration de la méthode à un seuil

On le voit, l'utilisation de deux seuils plutôt qu'un n'apporte rien. De plus, on le voit sur la figure 13, la cuve est souvent pleine, pour profiter de quelques pics dans les prix. Cela indique que cette stratégie n'est pas optimale, puisque sur une période où la cuve est pleine, il devrait être possible de déstocker un petit peu d'énergie lorsque le prix n'est pas trop faible, puis de la reremplir. Et de faire ainsi un petit profit supplémentaire.

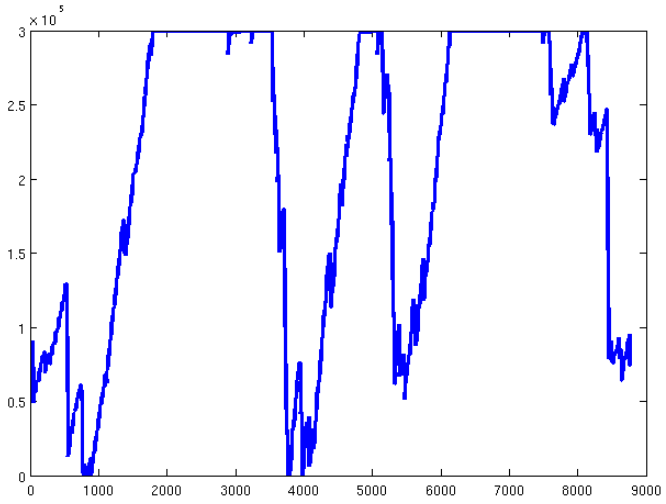

FIGURE 13 – Volume occupé dans la cuve en fonction du temps, pour deux seuils fixes

L'idée maintenant est de reprendre la méthode à un seul seuil, mais cette fois-ci de faire varier le seuil au cours du temps. On peut par exemple prendre comme seuil la moyenne des prix sur les K prochaines heures :

$$\mathrm{S}^n = \frac{1}{\mathrm{K}} \sum_{k=0}^{\mathrm{K}-1} p_x^{n+k}.$$

Cependant, une telle moyenne ne prend pas assez en compte les prix exceptionnels. En effet, si l'on voit venir à l'horizon un prix important, il vaut mieux se mettre à comprimer, pour pouvoir profiter au mieux, le moment venu, de ce pic des prix ; pourtant, une simple moyenne comme celle-ci ne relèvera probablement pas assez le seuil. Pour prendre en compte cela, on peut plutôt considérer

$$\mathrm{S}_q^n = \left[\frac{1}{\mathrm{K}} \sum_{k=0}^{\mathrm{K}-1} \left(p_x^{n+k} \right)^q \right]^{1/q}$$

Remarquons qu'ainsi, si $q \to \infty$, alors S_q^n tend vers le maximum du prix sur les K prochaines heures.

En procédant ainsi, pour $\mathrm{K} = 96$ h et $q = 30$, on obtient $8,1$ M ? de recettes. Le volume maximum occupé dans la cuve est alors de 170 000 m^3, sur les 300 000 disponibles.

3.4 Seconde amélioration de la méthode à un seuil

Au lieu de prendre une moyenne, on pourrait plutôt prendre une sorte de médiane. Ceci reviendrait à fixer $r \in \{1, ..., \mathrm{K}\}$, puis chercher à décomprimer au moins r fois dans une période de K heures, et à comprimer le reste du temps. Par exemple, on peut prendre

$$\mathrm{S}^n = r\text{-ième prix le plus élevé dans } \left\{ p_x^n, ..., p_x^{n+\mathrm{K}-1} \right\},$$

en comptant les multiplicités. Ainsi, si $\mathrm{K} = 4$ et $r = 2$, et que les prix dans les 4 prochaines heures sont $64, 100, 77, 100$, on prend $\mathrm{S} = 100$?. En procédant de la sorte, on obtient pour $\mathrm{K} = 96$ h et $r = 6$, « seulement » $8,0$ M ? de recettes.

Et si on faisait varier r ? Par exemple en fonction du taux de remplissage de la cuve ? De sorte à comprimer facilement lorsque le volume est faible, et décomprimer facilement lorsque la cuve est presque pleine. On peut donc prendre quelque chose comme

$$r^n = \mathrm{E}\left[\mathrm{K}\left(1 - \chi\left(1 - \frac{\mathrm{V}^n}{\mathrm{V}_{\max}}\right)\right)\right],$$

où $\mathrm{E}(x)$ représente la partie entière de x. Pour $\mathrm{K} = 96$ h et $\chi = 1,05$, on arrive alors à des recettes de $8,4$ M ?. Le volume maximal utilisé dans la cuve est alors seulement de $97\,000$ m^3.

4 Deux méthodes itératives

4.1 Placement progressif des décompressions

Il s'agit d'une stratégie qui, partant d'un scénario où on ne fait que compresser, place une par une les décompressions. Ces décompressions sont placées de sorte que :

- on place toujours la décompression à l'endroit où le prix est le plus élevé possible ;
- on fait en sorte qu'une décompression placée à l'instant n n'empiète pas sur les décompressions placées précédemment (et donc plus rentables).

Pour simplifier le problème, on a supprimé ici la contrainte que la compression doit durer au moins deux heures.

Dans ce paragraphe, $\alpha = (\alpha^n)_{1 \leqslant n \leqslant \mathrm{N}}$ et $\beta = (\beta^n)_{1 \leqslant n \leqslant \mathrm{N}}$ représentent les taux de compression et de décompression *souhaités*. Ils vérifient la contrainte $\alpha^k \beta^k = 0$ et comme auparavant, $\alpha^k, \beta^k \in [0, 1]$. On dispose d'une fonction qui calcule le volume dans la cuve au cours du temps et le gain à α et β donnés qui fonctionne comme suit :

- le volume initial dans la cuve est donné : $\mathrm{V}_0 = 80\,000$ m^3 ;
- si à l'instant k on souhaite décompresser, *i.e.* si $\beta^k > 0$, on effectue cette décompression dans la mesure du possible (s'il n'y a pas assez d'air dans la cuve, le taux effectif de décompression sera inférieur au taux souhaité α_k et éventuellement, il sera nul), et on revend l'énergie produite ;
- si à l'instant k on souhaite compresser, on vérifie que cela est possible, *i.e.* que $\alpha^k \mathrm{prod}^k \geqslant 5$ et qu'il y a suffisament de place dans la cuve pour effectuer la compression (pour des raisons de simplicité et parce que le volume d'air compressé en une heure est assez faible, on ne change pas le taux effectif de compression) ;
- si on ne peut pas effectuer la compression souhaitée, ou si on ne souhaite ni compresser ni décompresser, on revend toute l'énergie produite.

On présente maintenant l'algorithme itératif qui permet de placer les décompressions une par une. Les entiers en indice correspondent au numéro de l'itération. On initialise l'algorithme avec $\alpha_0 = (1, \cdots, 1)$ et $\beta_0 = (0, \cdots, 0)$, ce qui correspond à un scénario où on souhaite compresser autant que possible. La première itération consiste à placer une décompression à l'instant où le prix de revente est le plus élevé.

Supposons maintenant α_k et β_k donnés. Il s'agit de placer une décompression à un instant propice. Pour cela, on introduit deux paramètres $\mathrm{V}_{\text{seuil}}$ et `délai`. Les heures n possibles sont celle où l'une où l'autre des deux conditions suivantes est remplie :

– $V_k^n \geqslant 2V_{\text{seuil}}$, ce qui signifie qu'il y a suffisament d'air dans la cuve pour encaisser deux décompressions coup sur coup. Par exemple, en disant grossièrement qu'en une heure de décompression, la cuve se vide de 8 000 m^3, on peut prendre $V_{\text{seuil}} =$ 8 000 m^3. Cela permet d'assurer que si on avait placé une décompression à l'instant n_0, en en placant une nouvelle à l'instant $n_0 - 1$ il restera assez d'air pour garder une décompression complète à l'instant n_0 (ce qui semble nécessaire, puisque cette décompression est plus rentable) ;
– si $\beta_k^{n_0} > 0$ et si $V_k^{n_0} < V_{\text{seuil}}$, on exclut le nombre d'heures correspondant au paramètre délai avant n_0. L'idée est de ménager un temps à la cuve pour se remplir avant les décompressions qui ont lieu lorsque la cuve est presque vide. Par exemple, si $V_{\text{seuil}} = 8\,000$ m^3 et qu'il faut 50 heures pour compresser 8 000 m^3 d'air, on peut prendre `délai` $= 50$.

On place ensuite la décompression au moment n_1 où le prix de revente de l'énergie est le plus élevé parmis ces heures autorisées. Finalement, $\alpha_{k+1} = \alpha_k$ sauf en n_1 : $\alpha_{k+1}^{n_1} = 0$ et $\beta_{k+1} = \beta_k$ sauf en n_1 : $\beta_{k+1}^{n_1} = 1$ (à l'instant n_1, on décompresse au lieu de compresser).

Une amélioration immédiate de cet algorithme consiste à s'offrir plusieurs choix à chaque itération. Le programme utilisé pour les simulations numériques compare quatre possibilités à chaque itération (une décompression complète, une à 70 % une à 40 % et une où au lieu de compresser, on ne fait rien). Les résultats numériques sont assez satisfaisants puisqu'on obtient des gains à peu près équivalents au meilleur modèle à deux seuils adaptatifs. L'allure du volume dans la cuve (figure 15) montre que l'on utilise ici pleinement la cuve. Notons toutefois que l'algorithme met bien plus de temps à tourner puisqu'il faut environ 1 200 itérations. De plus il y a au moins deux paramètres à régler (voir figure 16).

4.2 Améliorations possibles

Côté améliorations, on pourrait prendre en compte plus de possibilités à chaque itération, y compris certaines menant à des compressions d'une partie seulement de l'énergie produite, pas du tout considérée ici. Les tests pour trouver les heures qu'on s'autorise à modifier pourraient sans doute être améliorés. Par exemple, ils pourraient dépendrent plus fortement du scénario envisagé (le test $V_k^n \geqslant 2V_{\text{seuil}}$ semble trop fort si on ne veut décompresser qu'un petit volume). Enfin, plutôt que de garder un paramètre délai, il serait peut être intéressant de modifier le prix de revente à chaque itération. Ainsi, les heures qui précédent une décompression très rentable sont plus précieuses qu'il n'y parait puisqu'on a intérêt à remplir la cuve. Ainsi le bénéfice attaché à une décompression pourrait être distribué sur les heures qui la précède (mais comment ?).

4.3 Utilisation d'une fonction « envie de (dé)compresser »

Lors des simulations à deux seuils fixes, on observe l'apparition de longues périodes de temps où la cuve reste pleine. Essayons d'introduire une fonction Envie signifiant à peu près que lors de telles périodes, on est plus enclin à effectuer des décompressions. Un premier essai consiste à introduire l'envie de décompresser

$$\text{ED}(p_x, \text{V}) = \frac{p_x \frac{\text{V}}{\text{V}_{\max}}}{pxd\left(1 - \frac{\text{V}}{\text{V}_{\max}}\right)}$$

ainsi que l'envie de compresser

$$\mathrm{EC}(p_x, \mathrm{V}) = \mathbf{1}_{\mathrm{prod}\geqslant 5} \frac{pxd\left(1 - \frac{\mathrm{V}}{\mathrm{V_{max}}}\right)}{p_x \frac{\mathrm{V}}{\mathrm{V_{max}}}}$$

Ici, pxd et pxc sont des paramètres qui jouent le rôle de seuil lorsque la cuve n'est ni presque vide ni presque remplie. De plus on a :

- l'envie de décompresser est d'autant plus forte que la cuve est pleine et nulle quand elle est vide :

$$\lim_{\mathrm{V}\to\mathrm{V_{max}}} \mathrm{ED}(p_x, \mathrm{V}) = +\infty \quad \text{et} \quad \lim_{\mathrm{V}\to\mathrm{V_{max}}} \mathrm{ED}(p_x, \mathrm{V}) = 0;$$

- l'envie de compresser est d'autant plus faible que la cuve est pleine et nulle quand elle est remplie :

$$\lim_{\mathrm{V}\to\mathrm{V_{max}}} \mathrm{EC}(p_x, \mathrm{V}) = 0 \quad \text{et} \quad \lim_{\mathrm{V}\to\mathrm{V_{max}}} \mathrm{EC}(p_x, \mathrm{V}) = +\infty.$$

Cette fonction permet de remplacer les seuils en utilisant par exemple un algorithme du type

- si $\mathrm{ED}(p_x^k, \mathrm{V}^k) - \mathrm{EC}(p_x^k, \mathrm{V}^k) > 1$ on décompresse à l'instant $k+1$;
- si $\mathrm{ED}(p_x^k, \mathrm{V}^k) - \mathrm{EC}(p_x^k, \mathrm{V}^k) < -1$ on compresse à l'instant $k+1$;
- sinon on ne fait rien.

On a encore deux paramètres à choisir, sans parler de la forme de la fonction (on pourrait ainsi envisager de mettre des puissances plus importantes type $(\mathrm{V}/\mathrm{V_{max}})^n$ pour favoriser les cas extrêmes). On augmente ainsi légèrement le bénéfice observé avec la méthode à deux seuils fixes.

On peut essayer de se servir de cette fonction pour construire une méthode itérative. On part de α^0 et β^0 donnés (par exemple, tout les deux nuls : on ne se sert pas de la cuve). Etant donnés α^k et β^k, on calcule le volume dans la cuve au cours du temps V^k puis les fonctions EC et ED correspondant à ce volume. On en déduit α^{k+1} et β^{k+1} de la même manière qu'exposée précédemment. Le choix des fonctions EC et ED ci-dessus n'est plus judicieux. En effet, à cause des explosions en $\mathrm{V} = 0$ et en $\mathrm{V} = \mathrm{V_{max}}$, un palier où la cuve est vide se transforme à l'itération suivante en un palier où la cuve est pleine. Il semble plus judicieux de prendre

$$\mathrm{ED} = \frac{p_x}{pxd} f(\mathrm{V})$$

avec f affine par morceaux (par exemple trois morceaux : un correspondant à la cuve presque vide, un à la cuve presque pleine et un à la cuve « normalement remplie"). Avec un tel choix, on a bien convergence de l'algorithme, mais les valeurs ne sont pas très intéressantes par rapport aux autres stratégies. Cela vient peut être du fait qu'on reste finalement assez proche de la stratégie à deux seuils fixes avec ces choix. Une piste intéressante est de modifier a posteriori l'envie de compresser de sorte à anticiper les décompressions les plus rentables. On peut par exemple augmenter significativement l'envie de compresser dans les heures précédant les décompressions. Le problème est de choisir comment on effectue cette modification et sur quel intervalle. Quelques tatonnements informatiques ont montré que cela pouvait en effet permettre d'augmenter le bénéfice mais que les choix à effectuer n'apparaissaient pas très clairement. Enfin, les taux de compression et de décompression sont

entièrement modifiés d'une itération à l'autre, ce qui ne semble pas très malin. Il pourrait être intéressant de ne modifier ces taux que là où ED et EC sont très élevés afin d'avoir effectivement un scénario qui s'améliore petit à petit. Pour conclure, cet algorithme n'est probablement pas bien adapté au problème. Il y a beaucoup de paramètres à régler et les modifications effectuées à chaque itération sont trop grossièrement choisies. De plus et en l'état, il n'apporte pas d'améliorations significatives par rapport à d'autres modèles plus simples.

5 Pistes pour optimisation globale

On peut aussi envisager le problème plus globalement, en fixant d'abord sur l'année notre comportement pendant certaines heures. On adapte ensuite le reste de nos actions pour rentabiliser au mieux le choix précédent. L'approche présentée ici pour décider du reste de ces actions est centrée sur la construction de la courbe de volume au cours de l'année.

5.1 Choix préalable des plages de décompression

On choisit ici de fixer initialement la liste des heures de décompression.

Dans un premier temps on décrit dans quel ordre on décide de traiter les différentes plages horaires séparant les plages de décompression. On utilisera une liste des heures de décompression mise à jour au cours de l'algorithme.

1. Dans un premier temps on souhaite fixer les heures les plus rentables pour décompresser. Pour cela on peut :
   (a) soit fixer un nombre N d'heures et choisir les N heures où le prix de vente est le plus élevé,
   (b) soit fixer un seuil minimum et choisir les heures où le prix est supérieur au seuil.

   Dans les deux cas on peut optimiser le paramètre ultérieurement, et on obtient une famille d'heures $\mathcal{F}_{\mathcal{N}} = \{t^n\}_{n\in\mathcal{N}}$ de déstockage ;
2. On souhaite ensuite fixer les plages de compression. Pour cela, on traite d'abord le créneau séparant les deux heures les plus rentables, $(t_0, t_1) \in \mathcal{F}^2_{\mathcal{N}}$, et on retire de la liste des heures de déstockage à traiter toutes celles qui se trouvaient entre les deux, i.e. $t^n \in \mathcal{F}_{\mathcal{N}}$ telles que $t^n \in [t_0, t_1]$;
3. Dans la liste mise à jour, on choisit l'heure la plus rentable t^j et on traite l'intervalle qui la sépare des intervalles de temps traités précédemment,
   (a) si $t^j > t_1$ alors on traite l'intervalle $[t_1, t^j]$,
   (b) si $t^j < t_0$ alors on traite l'intervalle $[t^j, t_0]$,

   on retire de la liste les heures contenue dans l'intervalle traité, puis on recommence jusqu'à atteindre la fin de la liste.

Pour compléter le comportement sur l'année entière il restera alors à décompresser le plus possible avant la fin de l'année d'une part - donc entre la dernière des décompression de $\mathcal{F}_{\mathcal{N}}$ et la dernière heure de l'année - et au début de l'année d'autre part - donc entre la première heure de l'année et la première décompression de $\mathcal{F}_{\mathcal{N}}$. Cela fixera d'ailleurs le volume initial.

Dans un second temps on décrit le traitement d'un intervalle de temps $[t_\alpha, t_\beta]$.

1. On impose le volume stocké à sa valeur maximale au début de la plus rentable de ces deux bornes
$$t \in \{t_\alpha, t_\beta\} \text{ tel que } p_x(t) = max\{p_x(t_\alpha), p_x(t_\beta)\}. \quad (1)$$
On appellera cette borne la première, l'autre la seconde ;
2. On utilise ensuite les plages horaires les plus rentables entre les deux pour stocker, afin d'obtenir le plus grand volume possible au début de la seconde borne, dans un premier temps sans tenir compte des décompressions initialement prévues entre les deux heures ;
3. Parmi ces décompressions initialement prévues, on impose la plus rentable, puis on vérifie s'il est possible de compresser pendant assez de plages horaires avant et après elle pour ne pas dégrader la valeur du volume au début de la seconde borne ;
4. Si c'est le cas, on la garde, on l'enlève de la liste puis on recommence avec la liste mise à jour ; si ce n'est pas le cas, on annule son effet puis de la même manière on l'enlève de la liste et on recommence avec la liste mise à jour ;
5. Lorsque toutes les heures de décompression initialement prévues entre t_α et t_β ont été traitées, on considère que la plage $[t_\alpha, t_\beta]$ est traitée.

Remarque : on pourrait également procéder en inversant les heures de compression avec les heures de décompression, et comparer les résultats obtenus dans les deux cas.

L'idée était ici de choisir, sur une plage horaire restreinte, les heures de compression les plus rentables. On voit donc qu'une autre possibilité serait de traiter également les heures de compression de manière globale.

5.2 Choix préalable des plages prioritaires de compression et de décompression

L'idée serait de de traiter de manière plus équilibrée les deux comportements de stockage et de déstockage. On choisirait alors de fixer la liste de la totalité des heures où l'on souhaite décompresser, ainsi que la liste de la totalité des heures où l'on souhaite décompresser.

Comme dans la méthode précédente, il faut à nouveau choisir un critère pour établir ces deux listes. On peut ici aussi utiliser des valeurs seuils ou des nombres d'heures. Cette méthode est en fait la méthode mise en place par l'équipe de GDF.

Remerciements

Nous tenons à remercier chaleureusement les organisateurs de cette première Semaine d'Étude Maths-Entreprises, pour nous avoir donné l'opportunuité d'explorer des thématiques nouvelles dans un univers différent de celui de la recherche académique, ainsi qu'Alain Fuser et, plus généralement, GDF–Suez, pour avoir bien voulu jouer le jeu. Naturellement, nous voulons aussi remercier tous les mathématiciens qui ont pu nous aider, par leurs conseils ou leurs suggestions, et notamment Laurent Dumas, Bertrand Maury, Gabriel Peyré et Filippo Santambrogio. Enfin, nous remercions l'IHP d'avoir bien voulu nous prêter les locaux où travailler.

Références

[1] G. Allaire et G. Michailidis : Conception de pièces massives par optimisation topologique avec pris en compte de contraintes de fabrication. Thèse en cours.

[2] L. Evans : *Partial Differential Equations*, volume 19 de *Graduate Studies in Mathematics*. American Mathematical Society, 1998.

[3] A. Henrot et M. Pierre : *Variation et Optimisation de Formes*, volume 48 de *Mathématiques et Applications*. Springer, 2005.

[4] S. Osher et R. Fedkiw : *Level set methods and dynamic implicit surfaces*, volume 153 de *Applied Mathematical Sciences*. Springer-Verlag, New York, 2003.

[5] S. Osher et J.A Sethian : Front propagation with curvature dependent speed : algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 78:12–49, 1988.

[6] R. Tyrrel Rockafellar et Roger J-B Wets : *Variational Analysis*. Springer, 1998.

[7] H.-K. Zhao, T. Chan, B. Merriman et S. Osher : A variational level set approach to multiphase motion. *J. Comput. Phys.*, 127(1):179–195, 1996.

[8] G. Zoutendijk : Methods of feasible directions. *Elsevier Publishing Company*, 1960.

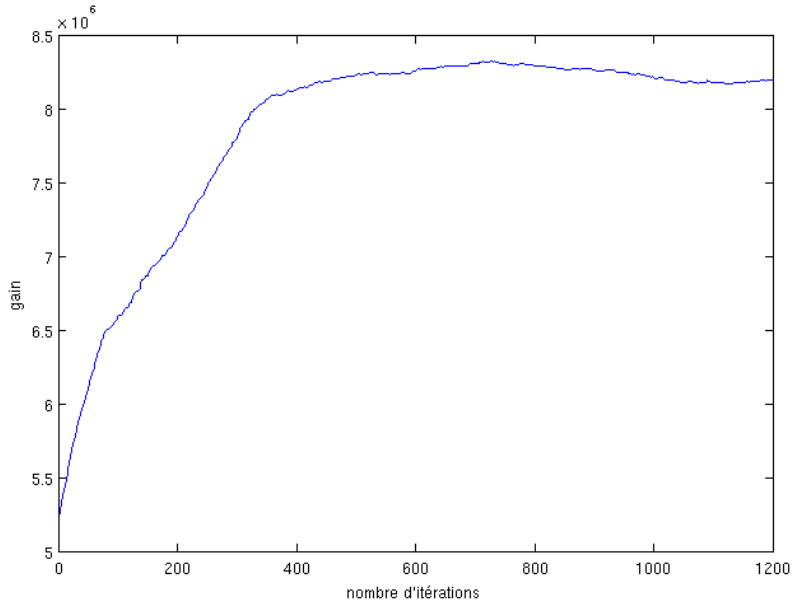

FIGURE 14 – Evolution du gain au cours des itérations

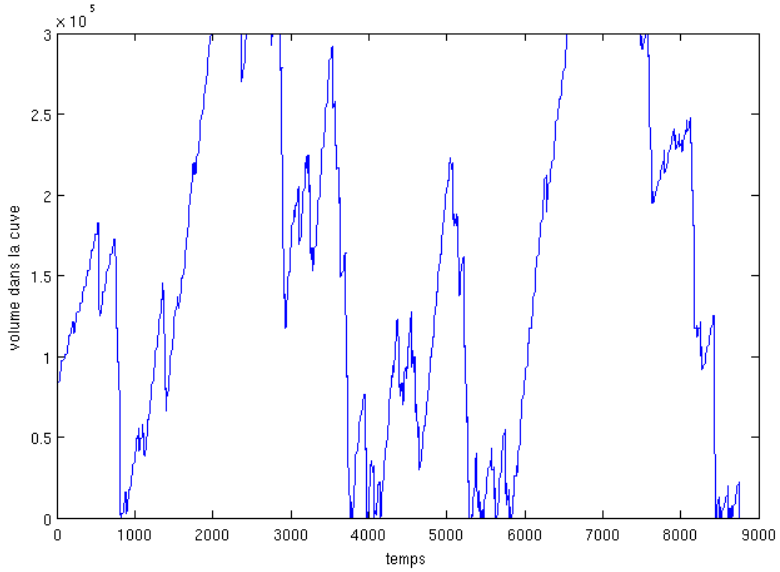

FIGURE 15 – Volume dans la cuve au cours du temps avec la méthode de placement progressif des décompressions.

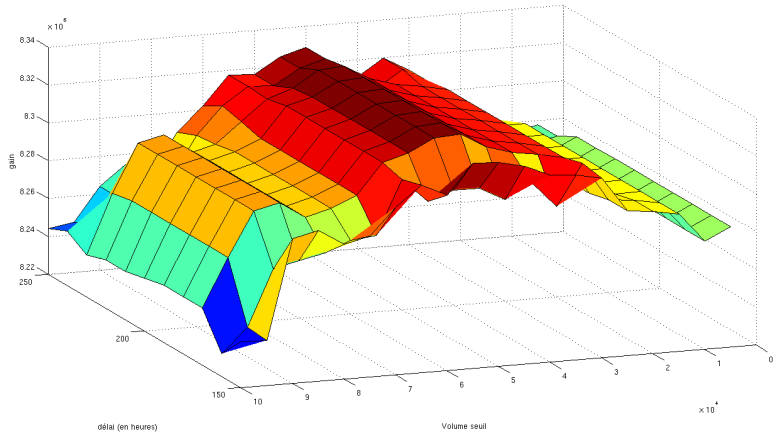

FIGURE 16 – Gain en foncion des paramètres pour la méthode de placement progressif des décompressions.